OPTIMIZING UPDATABLE SCROLLABLE CURSORS IN DATABASE SYSTEMS

FIELD OF THE INVENTION

The present invention is directed to an improvement in computing systems and in particular to optimizing command execution in computer database systems that provide support for updatable scrollable cursors.

BACKGROUND OF THE INVENTION

The Open Database Connectivity (ODBC) specification supports updatable scrollable cursors for Relational Database Management Systems (RDBMSs). This standard provides that multiple cursors may be defined for tables in relational databases and that positioned UPDATEs and DELETEs may be performed on the tables based on the scrollable cursor's location. The ODBC standard also provides for an attribute in the database to define an optimistic concurrency scheme. In the ODBC standard this attribute is referred to as SQL_CONCUR_VALUES. Use of the SQL_CONCUR_VALUES attribute provides that a positioned UPDATE or DELETE succeeds only if the record data to be modified has not been changed since it was last fetched by the user.

In certain relational database systems such as the DB2 UDB (trade-mark) RDBMS, for each scrollable cursor a temporary copy of record data is made when the data is fetched by the user. Where the SQL_CONCUR_VALUES attribute is applied, and a positioned UPDATE or DELETE is to be carried out, the temporary copy of record data must be compared to the current record data in the database to ensure that the record data has not changed since the time that it was copied to the temporary location. A comparison of the

record in the temporary copy with the record in the current table may result in significant overhead cost for the UPDATE or DELETE where the records to compare are extensive.

It is therefore desirable to have a relational database system that will support the ODBC updatable scrollable cursors and the SQL_CONCUR_VALUES attribute in which it is possible to optimize the steps to carry out the positioned UPDATE or DELETE commands.

SUMMARY OF THE INVENTION

A system, method and computer readable medium containing programming instructions for optimizing command execution in a database system is disclosed. According to one embodiment of the present invention, the database system stores data records on data pages, and provides a log sequence number for each data page. The log sequence number indicates when any of the data records contained in the data page were last modified. A data record is selected from a data page and copied to a second storage area. The present invention verifies that the selected data record has not been modified since the time that is was copied to the second storage area based upon the log sequence number, and then executes the command.

Advantages of the present invention include a reduction in processing time of database UPDATE or DELETE operations based on scrollable cursors where there is support for optimistic concurrency.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram representing example tables in a database subject to the

optimization of the preferred embodiment.

In the drawings, the preferred embodiment of the invention is illustrated by way of example. It is to be expressly understood that the description and drawings are only for the purpose of illustration and as an aid to understanding, and are not intended as a definition of the limits of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Figure 1 illustrates, in a block diagram, data which is subject to the optimization of the preferred embodiment. Figure 1 shows a portion of a relational database containing rows m and n, shown as records 12, 14 in the figure. Figure 1 also shows a temp table 16 in which data corresponding to rows m and n are shown as records 18, 20.

In the preferred embodiment, records in the relational database are stored on pages. Each page has a log sequence number (LSN) associated with the page. In Figure 1, data page 10 is shown with associated LSN 22. The LSN for the data page includes information which effectively provides a time stamp of the last modification made to any table data (records) on that page. With reference to the example of Figure 1, LSN 22 is updated when either record 12 or record 14, or any other record stored on data page 10, is modified in any way.

As is indicated in the diagram of Figure 1, according to the preferred embodiment, a copy of the data page LSN is maintained in association with a record when that record is written to temp table 16. Temp table 16 is used to copy row values when an updatable scrollable cursor is used to retrieve table record values for a user. In the example of Figure 1, a cursor has been used to access rows m and n and therefore the row m value in record 12

in data page 10 is copied to record 18 in temp table 16. Similarly the row n value in record 14 is copied to temp table 16 record 20 when a cursor in table is used to fetch row n for a user.

In the preferred embodiment, when record 18 having the value of row m is stored in temp table16, a copy of LSN 22 is made and stored in the temp table in association with record 18. This is shown in Figure 1 as LSN 24. Similarly, a copy of LSN 22 is made in association with record 20 when the value of row n is copied to temp table 16. This associated LSN value is shown as LSN 26 in Figure 1.

Due to the concurrency available in the ODBC standard, it is possible for row m to be copied to temp table 16 from data page 10 with the then current value of a LSN 22 being copied to LSN 24 and to then have a subsequent modification to data page 10 before the cursor reaches row n (in the table record 14). As a result, row n values may be copied into record 20 in temp table 16 with LSN value 26 that differs from LSN 24. This is due to a change to the value of LSN 22 when the data page 10 values are modified prior to copying the value of row n into temp table 16.

In the preferred embodiment, where SQL_CONCUR_VALUES attribute is associated with the table containing data page 10, and an UPDATE or DELETE operation is specified for, for example, row m or row n, based on the position of a cursor, it is necessary to ensure that the value of rows m and n as stored in temp table 16 are the same as the values as stored in data page 10. Although this may be carried out by a direct comparison of the attribute values in the respective rows in temp table 16 and data page 10, the preferred embodiment is able to potentially avoid such a direct comparison by comparing the LSN values of the records. Where, for example, row m is subject to a positioned UPDATE or

DELETE based on a scrollable cursor, the value of LSN 24 is compared with the value of LSN 22. Where these values match, it is necessarily the case that record 12 has not been modified since a copy of the value of row m was copied into record 18 in temp table 16. If record 12 had been modified, the value of LSN 22 would have changed and there would therefore not be a match between the value of LSN 22 and the value of LSN 24. In this manner, it is possible to avoid comparing all attribute values of record 12 and of record 18 by carrying out the simple comparison of the value of LSN 24 and the value of LSN 22. Where these values are different, it will be necessary to carry out the direct comparison of record 12 and record 18. However, this step may be avoided where the values of LSN 24 and LSN 22 are found to match. Where this is the case, the user will be able to directly update the value of record 12 and rely on the record 18 value as corresponding to record 12.

In the implementation of the preferred embodiment, the retrieval and comparison of LSN values is carried out using an internal command referred to as FETCH SENSITIVE NO DATA. The operations carried out by the FETCH SENSITIVE NO DATA internal command result in the LSN for the data page and the LSN for the record in the temp table being retrieved and compared, as is described above. In this way, the RDBMS of the preferred embodiment is able to execute the FETCH SENSITIVE NO DATA command as a part of the steps taken by the RDBMS in optimizing execution of UPDATE or DELETE commands in response to a user request.

The preferred embodiment is described above with respect to the UPDATE and DELETE commands where a RDBMS supports optimistic concurrency for a scrollable cursor. The optimization of the preferred embodiment may also be used in implementing other commands in an RDBMS which include a temporary table copy of a record, and

require a confirmation that the temp table copy is equivalent to the database copy.

An example of such an implementation of the optimization of the preferred embodiment is with respect to the FETCH SENSITIVE command in the DB2 UDB ™ RDBMS. The FETCH SENSITIVE command is available to users (in contrast to the FETCH SENSITIVE NO DATA command referred to above which is used internally in the RDBMS, only). Execution of the command, without any optimization, results in the fetch of a record from the database table and the qualification of that record (its attribute values are compared with the SQL predicates associated with the command). Where the record qualifies, the temp table is updated and the record is returned to the user.

The optimization of the preferred embodiment makes it possible to use the value of the LSN stored in the temp table to avoid steps in carrying out the command. Where the row m, for example, has been previously fetched and is in temp table 16, and the value of LSN 24 is equivalent to data page 10 LSN 22, a FETCH SENSITIVE command carried out on row m may be implemented by positioning the cursor at the appropriate record and returning a flag to the user to indicate that the previously fetched values remain current. If the two LSN values are not equal, then the non-optimized steps to carryout the FETCH SENSITIVE command are followed.

Using the comparison of the LSN value associated with the temp table record and the LSN value of the data page, the copying of attribute values to the temp table may be avoided. In this manner, the optimization of the preferred embodiment may be used to increase efficiency in carrying out command execution in a database with updatable scrollable cursors.

Although a preferred embodiment of the present invention has been described here in

detail, it will be appreciated by those skilled in the art, that variations may be made thereto, without departing from the spirit of the invention or the scope of the appended claims.